AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of logging and recovery in a transaction-processing system having a main memory for storing a database, wherein the database comprises a plurality of objects, any of which is denoted by b hereafter, one or more persistent backup storage devices for storing a copy of the database in the main memory, and one or more persistent log storage devices for storing log records,

the logging method comprising:

generating a differential log record using $\Delta_t = b_{t-1} \oplus b_t$ where Δ_t is the differential log record generated for the t-th update on a database object b, $\oplus$ is the bit-wise exclusive-OR (XOR) operation, b_{t-1} is the image of the database before the t-th update occurs, and b_t is the image of the database after the t-th update occurs; and

the recovery method comprising:

redoing updates of committed transactions using $b_{s+p} = b_s \oplus \Delta_{(1)} \oplus \Delta_{(2)} \ldots \oplus \Delta_{(p)}$, where b_s is the image of a database object b after s number of updates are reflected, b_{s+p} is the image of b rolled forward from b_s by p number of updates involved in committed transactions, $\oplus$ is the bit-wise XOR operation, and where the sequence of the differential log records applied, $\Delta_{(1)}$ through $\Delta_{(p)}$, is any possible arrangement of the set of sequentially-generated differential log records $\{\Delta_{s+1}, \ldots , \Delta_{s+p-1}, \Delta_{s+p}\}$ in the order independent from the order of log creation, and

undoing updates of uncommitted transactions using $b_{s-q} = b_s \oplus \Delta_{(1)} \oplus \Delta_{(2)} \ldots \oplus \Delta_{(q)}$, where b_s is the image of a database object b after s number of updates are reflected, b_{s-q} is the image of b rolled backward from b_s by q number of updates involved in uncommitted transactions, $\underline{\oplus \text{ is the bit-wise XOR operation,}}$ and where the sequence of the differential log records applied, $\Delta_{(1)}$ through $\Delta_{(q)}$, is any possible arrangement of the set of sequentially-generated differential log records $\{\Delta_{s-q+1}, \Delta_{s-q+2}, \ldots , \Delta_s \}$ in the order independent from the order of log creation.

2. (Original) The method of Claim 1, wherein the database comprises a plurality of fixed-size pages.

3. (Previously Presented) The method of Claim 2, wherein said each log record has a log header comprising:

LSN (Log Sequence Number) for storing a log sequence;

TID(Transaction ID) for storing the identity of the transaction that created the log record;

Previous LSN for storing the identity of the most recently created log by the same transaction;

Type for storing the type of the log record;

Backup ID for storing the relation between the log record and the updated page for use with fuzzy checkpointing;

Page ID for storing the identity of an updated page;

Offset for storing the starting offset of an updated area within the updated page; and

Size for storing the size of the updated area.

4. (Previously Presented) The method of Claim 1, further comprising:

checkpointing by occasionally writing the database in the main memory to said one or more persistent back storage devices.

5. (Previously Presented) The method of Claim 4, wherein checkpointing uses a transaction consistent checkpointing policy.

6. (Previously Presented) The method of Claim 4, wherein checkpointing uses an action consistent checkpointing policy.

7. (Previously Presented) The method of Claim 4, wherein checkpointing uses a fuzzy checkpointing policy.

8. (Previously Presented) The method of Claim 4, wherein the recovery method further comprises:

loading the checkpointed database from said one or more persistent backup storage devices into the main memory database; and

loading the log records from said one or more persistent log storage devices into the main memory database in order to restore the main memory database to the most recent consistent state.

9. (Previously Presented) The method of Claim 8, wherein loading the checkpointed database is executed in parallel by partitioning data in said one or more backup storage devices.

10. (Previously Presented) The method of Claim 8, wherein the recovery method is done in two passes by separating a redoing pass and an undoing pass.

11. (Previously Presented) The method of Claim 10, wherein reading the log records and redoing/undoing the log records are executed in a pipeline.

12. (Previously Presented) The method of Claim 10, wherein reading the log records is executed in parallel by partitioning the log records as well as redoing/undoing the log records.

13. (Previously Presented) The method of Claim 12, wherein reading the log records and redoing/undoing the log records are executed in a pipeline.

14. (Previously Presented) The method of Claim 8, wherein redoing/undoing the log records is done in one pass.

15. (Previously Presented) The method of Claim 14, wherein reading the log records and redoing/undoing the log records are executed in a pipeline.

16. (Previously Presented) The method of Claim 14, wherein reading the log records and redoing/undoing the log records are executed in parallel by partitioning the log records.

17. (Previously Presented) The method of Claim 16, wherein reading the log records and redoing/undoing the log records are executed in a pipeline.

18. (Previously Presented) The method of Claim 8, further comprising filling the main memory database with 0s in advance.

19. (Previously Presented) The method of Claim 18, wherein loading the checkpointed database comprises:

reading the checkpointed database from said one or more backup storage devices; and

redoing/undoing the checkpointed database by applying the XOR operation between the checkpointed database and the main memory database.

20. (Previously Presented) The method of Claim 19, wherein reading the checkpointed database and redoing/undoing the checkpointed database are executed in a pipeline.

21. (Previously Presented) The method of Claim 19, wherein reading the checkpointed database is executed in parallel by partitioning the checkpointed database as well as redoing/undoing the checkpointed database.

22. (Previously Presented) The method of Claim 21, wherein reading the checkpointed database and redoing/undoing the checkpointed database are executed in a pipeline.

23. (Previously Presented) The method of Claim 19, wherein loading the checkpointed database and loading the log records are executed in parallel.

24. (Currently Amended) A transaction processing system allowing logging updates and recovering from failure, comprising:

a main memory for storing a database;

one or more persistent log storage devices for storing log records;

one or more persistent backup storage devices for storing a copy of the database in the main memory;

means for generating a differential log record using $\Delta_t = b_{t-1} \oplus b_t$ where Δ_t is the differential log record generated for the t-th update on a database object b, b_{t-1} is the before-update image and b_t is the after-update image, and $\oplus$ is the bit-wise XOR operation; and

means for replaying the differential log records in an arbitrary order, independent of their generation order, by using the bit-wise XOR operations,

wherein said means for replaying further comprises:

means for redoing committed transactions using $b_{s+p} = b_s \oplus \Delta_{(1)} \oplus \Delta_{(2)} \ldots \oplus \Delta_{(p)}$, wherein b_s is the image of a database object b after s number of updates are reflected, b_{s+p} is the image of b rolled forward from b_s by p number of updates involved in committed transactions, and $\oplus$ is the bit-wise XOR operation, where the sequence of the differential log records applied, $\Delta_{(1)}$ through $\Delta_{(p)}$, is any possible arrangement of the set of sequentially-generated differential log records $\{\Delta_{s+1}, \ldots, \Delta_{s+p-1}, \Delta_{s+p}\}$ in the order independent from the order of log creation; and

means for undoing uncommitted transactions using $b_{s-q} = b_s \oplus \Delta_{(1)} \oplus \Delta_{(2)} \ldots \oplus \Delta_{(q)}$, wherein b_s is the image of a database object b after s number of updates are reflected, b_{s-q} is the image of b rolled backward from b_s by q number of updates involved in uncommitted transactions, and $\oplus$ is the bit-wise XOR operation, where the sequence of the differential log records applied, $\Delta_{(1)}$ through $\Delta_{(q)}$, is any possible arrangement of the set of sequentially-generated differential log records $\{\Delta_{s-q+1}, \Delta_{s-q+2}, \ldots, \Delta_s\}$ in the order independent from the order of log creation.

25. (Original) The system of Claim 24, wherein the database comprises a plurality of fixed-size pages.

26. (Previously Presented) The system of Claim 24, further comprising:

means for checkpointing the database by occasionally writing the database in the main memory to one or more persistent backup storage devices.

27. (Previously Presented) The system of Claim 26, wherein the means for checkpointing uses a transaction consistent checkpointing policy.

28. (Previously Presented) The system of Claim 26, wherein the means for checkpointing uses an action consistent checkpointing policy.

29. (Previously Presented) The system of Claim 26, wherein the means for checkpointing uses a fuzzy checkpointing policy.

30. (Previously Presented) The system of Claim 26, wherein the means for replaying comprises:

means for loading the checkpointed database into the main memory database; and

means for loading the log into the main memory database.

31. (Previously Presented) The system of Claim 30, wherein the means for loading the checkpointed database comprises:

means for reading the checkpointed database from one or more persistent backup storage devices; and

means for playing the checkpointed database to restore the main memory database to the state when the backup was made by applying the XOR operations between the checkpointed database and the main memory database.

32. (Previously Presented) The system of Claim 30, wherein the means for loading the log comprises:

means for reading the log records from the persistent log storage devices; and

means for playing the log records in two passes to restore the main memory database to the latest consistent state.

33. (Previously Presented) The system of Claim 30, wherein the means for loading the log comprises:

means for reading the log records from the persistent log storage devices; and

means for playing the log records in one pass to restore the main memory database to the latest consistent state.

34. (Currently Amended) A computer-readable storage medium that contains a program for logging updates and recovering from failure in a transaction-processing system having a main memory for storing a database, one or more persistent backup storage devices for storing a copy of the database in the main memory, and one or more persistent log storage devices for storing log records, where the program under the control of a CPU performs:

generating differential log records by using $\Delta_t = b_{t-1} \oplus b_t$, where $\underline{\Delta_t}$ is the differential log record generated for the t-th update on a database object b, b_{t-1} is the before-update image and b_t is the after-update image, and $\oplus$ is the bit-wise XOR operation;

replaying the differential log records in an arbitrary order, independent of their generation order, by using the bit-wise XOR operations,

wherein replaying the differential log records further comprises:

redoing committed transactions using $b_{s+p} = b_s \oplus \Delta_{(1)} \oplus \Delta_{(2)} \ldots \oplus \Delta_{(p)}$, wherein b_s is the image of a database object b after s number of updates are reflected, b_{s+p} is the image of b rolled forward from b_s by p number of updates involved in committed transactions, and $\oplus$ is the bit-wise XOR operation, where the sequence of the differential log records applied, $\Delta_{(1)}$ through $\Delta_{(p)}$, is any possible arrangement of the set of sequentially-generated differential log records $\{\Delta_{s+1}, \ldots , \Delta_{s+p-1}, \Delta_{s+p}\}$ in the order independent from the order of log creation; and

undoing uncommitted transactions using $b_{s-q} = b_s \oplus \Delta_{(1)} \oplus \Delta_{(2)} \ldots \oplus \Delta_{(q)}$, wherein b_s is the image of a database object b after s number of updates are reflected, b_{s-q} is the image of b rolled backward from b_s by q number of updates involved in uncommitted transactions, and $\oplus$ is the bit-wise XOR operation, where the sequence of the differential log records applied, $\Delta_{(1)}$ through $\Delta_{(q)}$, is any possible arrangement of the set of sequentially-generated differential log records $\{\Delta_{s-q+1}, \Delta_{s-q+2}, \ldots , \Delta_s \}$ in the order independent from the order of log creation.

35. (Original) The storage medium of Claim 34, wherein the medium is a CD.

36. (Previously Presented) The storage medium of Claim 34, wherein the medium is a magnetic tape.

37. (Previously Presented) The method of Claim 1, further comprising one or more in-memory log buffers wherein each generated log record is temporarily stored in any available log buffer and a group of the buffered log records are written together to an arbitrary one of said one or more persistent log storage devices.